Efficiently identifying disguised nulls in heterogeneous text data

Théo Bouganim, Ioana Manolescu, Helena Galhardas

Efficiently identifying disguised nulls in heterogeneous text data

Théo Bouganim[★] Helena Galhardas[†] Ioana Manolescu[★]
[★]Inria & IPP, [†]INESC-ID & IST, Univ. Lisboa
theo.bouganim@inria.fr,hig@inesc-id.pt,ioana.manolescu@inria.fr

ABSTRACT

Digital data is produced in many data models, ranging from highly structured (typically relational) to semi-structured models (XML, JSON) to various graph formats (RDF, property graphs) or text. Most real-world datasets contain a certain amount of *null* values, denoting missing, unknown or unapplicable information. While some data models allow representing nulls by special tokens, so-called *disguised nulls* are also frequently encountered: these are values that are not syntactically speaking nulls, but which do, nevertheless, denote the absence, unavailability or unapplicability of the information.

This paper describes our ongoing work toward detecting disguised nulls in textual data, encountered in ConnectionLens graphs. Driven by journalistic applications, we focus for now on large, semistructured datasets, where most or all data values are free-form text. We show that the state-of-the-art methods for detecting nulls in relational databases, mostly tailored towards numerical data, do not detect disguised nulls efficiently on such data. Then, we present two alternative methods: (i) leveraging Information Extraction, and (ii) text embeddings and classification. We detail their performance-precision trade-offs on real-world datasets.

1 INTRODUCTION

Digital data is being produced and reused at unprecedented rates. Large datasets are usually processed within data management systems, which *model* the data according to a given data model, provide means to *store* it, and to *query* it using a declarative query language or another form of query API. The database industry has been pioneered by relational database management systems (RDBMSs), whose foundations lay in first-order logic, formalization by E. F. Codd [11] and subsequent work, e.g., [2].

Where there is data, there are null values Since the early database days, *nulls* have been identified as a central concept denoting missing, unknown or unapplicable information. The semi-structured data model, first embodied in OEM (the Object Exchange Model) allowed to hope the need for nulls would disappear: missing information is simply not represented at all in the data [15]. However, standard semi-structured models such as XML and JSON re-introduced, e.g., `xsi:nil` in tools supporting XML Schema or the special `null` value in JSON. Presumably, such null tokens were felt needed in practice because "perfect and complete" databases, regardless of their data model, are the exception rather than the norm. At query time, null values are typically handled through the so-called *three value logic*: a predicate over a null value always evaluates to *unknown* (neither *true* nor *false*), and query results only include tuples on which the query predicates evaluated to *true*.

Disguised nulls in relational databases In practice, it has been noted that relational databases often feature not only *null* tokens, but also *non-null values playing the semantic role of nulls*, also called *disguised nulls* [1]. For instance, the birth department of all French residents born out of France is coded 99 in public databases such as the Social Security one; 0 or -1 are often used to encode an unknown (but non-zero) number such as a price or a number of items sold; users may enter "none, "-", "unknown" or "N/A" or any other similar phrase or token, in entry forms requiring numbers, names or dates that they are unable or unwilling to fill in. Further, when a value needs to be chosen from a predefined set, such as a state of the U.S., users may forget to set it from the menu, leaving the default value which just happens to be the first, e.g., "Alabama" in a form requiring users to select a state of the USA.

Data entry forms sometimes prevent null codes by checking the entered value, e.g., "N/A" would not be accepted as a number. However, null codes may still persist: (*i*) users replace "N/A" with 0 for an unknown, non-zero number; (*ii*) if the expected input type is free text, e.g., "List of industrial collaborations in connection with this research", no simple format-driven validation applies; (*iii*) in cases such as "Alabama" above, the value is in the correct domain.

Detecting disguised nulls As explained above, null values require special treatment when querying data; this treatment is built in data management systems when the nulls are *explicit*. However, disguised nulls require dedicated detection methods, and several methods have been proposed for relational databases [17, 20, 21]. These methods are based on statistical analysis of the data. They can detect, for instance, when a value such as 0 is suspiciously frequent in a numeric attribute, or when a value of attribute $R.a$ is an outlier in the joint distribution of $(R.a, R.b)$, where we expect the distributions $R.a$, $R.b$ to be independent. Detecting disguised nulls is important for data cleaning (users may want to replace them with explicit nulls), and for query correctness: null values should not match any selection predicate, and there should be no join on null values.

Problem statement and outline In this work, we consider the detection of disguised nulls in *textual, heterogeneous data*. The motivation for our work came from ConnectionLens [5, 10], a system capable of integrating structured, semistructured or unstructured data into **graphs**, which are enriched by adding all the entities (people, organizations, places, URIs, dates etc.) encountered in various text nodes. We have developed ConnectionLens inspired by fact-checking and data journalism applications [6, 8]. We encountered many datasets where some fields are *free-form text* entered by users. For instance, in the French Transparency dataset HATVP, elected officials need to state "their direct financial participation in company capitals"; in the PubMed bibliographic database, free-form texts include the article titles, abstracts, funding statements, and

possible acknowledgments. Disguised nulls encountered in such fields range from "N/A" or "-" to "Liz Smith has not received any funding related to this work" or "No conflicts of interest, financial, or otherwise are declared by the authors." or "There is no conflict of interest relating to Authors. The manuscript was prepared according to scientific and ethical rules." (in this case, the financial acknowledgment question does not apply to the authors). Detecting disguised nulls in ConnectionLens graphs is important:

(1) First, it allows to avoid *unifying* nodes with identical or similar labels [5] if these are disguised nulls, e.g., if two companies are described as having "Unknown" CEO, this should not lead to a connection between the two companies through the node labeled "Unknown"; this can be seen as the graph counterpart of the fact that null values should not join in relational databases.
(2) Second, if we know that a string is a disguised null, we can avoid extracting entities from it; this is potentially useful, because entity extraction dominates by far the cost of constructing ConnectionLens graphs [5, 6].

How to detect such disguised nulls? We describe below alternative methods and simultaneously the structure of the paper.

(1) We first show that ConnectionLens entity extraction can be leveraged to (manually) establish an *entity profile* for each set of text attributes in which we want to detect nulls, and consider any value deviating from this profile a disguised null. For instance, the entity profile of financial participation descriptions could be *Organization+* to state that it should contain at least one organization, whereas the entity profile of a funding statement could be *Person+ Organization+*: in a funding acknowledgement, at least an author name and at least a funding organization should appear. This method is quite accurate, however, it incurs a high computational cost, since entity extraction is a complex operation (Section 4).
(2) To address this shortcoming, we devised a novel method, which relies on text embeddings and classification, while also leveraging entity extraction on a much smaller portion of the dataset.
(3) We demonstrate that this method is much more efficient than the one based on entity profiles, while also being very accurate (Section 5). We perform a set of experiments on the state of the art (FAHES) mostly aimed at numeric data. Our experiments show that they do not perform very well on free texts values and less structured data (Section 6.3).

We show the impact of our work on the time needed to build a ConnectionLens graph before concluding and providing some perspectives.

2 RELATED WORK

In this section, we recall the main methods for disguised null detection in relational database. To our knowledge, no similar methods have been studied for semi-structured nor graph data.

Foundational work in this area was made in [20], which introduced and formalized the problem of Disguised Missing Values (DMV), and measured its influence on different data science models. A set of *statistical models* (mutually disjoint hypothesis) were introduced in [19] on the existence of missing values (explicit or disguised nulls):

- The **MCAR** (Missing Completely At Random) model posits that the probability of a value to be missing is the same for any value of an attribute, and does not depend on the values of any other attribute. For instance, assuming an attribute is the result of a physical measure made with a device that breaks down, the resulting missing values are not correlated to any other aspect of the data or of the values.
- The **MAR** (Missing At Random) model considers that the probability for a value to be missing depends on values encountered in other attributes of the same table (these notions have been defined for tables). For example, in a political poll, let assume *young* voters are more likely not to declare their political preference. Then, the political preference value is MAR.
- **MNAR** (Missing Not At Random) applies when neither MAR nor MCAR hold, and when the probability for a value to be missing does depend on the actual value that is missing, but not on the values of any other attribute. For instance, assuming supporters of a certain political party generally avoid stating their preference and instead let that information go missing, such values are MNAR.

Building upon these models, [17] has proposed a heuristic method for identifying DMVs in relational databases. Under the MAR and MCAR assumptions, the authors assume that a value v in attribute A_i in a table T is a DMV if $\sigma_{A_i=v}(T)$ contains a subset $T^*_{A_i=v}$ that represents a good sampling of T. Such a subset is an *Embedded Unbiased Sample* (**EUS**) which means that except for attribute A_i, $T^*_{A_i=v}$ and T have similar distributions. Then, a **MEUS** (*Maximal EUS*) is intuitively an EUS with a good trade-off between size (larger is better) and similarity (in distribution) with T. largest EUS with the highest similarity. The gist of the [17] heuristics is to find MEUS in a dataset, and consider their associated $A_i = v$ values as DMVs.

FAHES [21] incorporates the method of [17], to which the authors add two other methods, in order to distinguish three classes of DMV.

- The first class contains **syntactic outliers**. A syntactic outlier is a value whose syntax is significantly different from that of other values in the same attribute. Two techniques are used to identify them. (*i*) Syntactic pattern discovery infers a frequent syntactic pattern (shape) for the values of each attribute, and points out the values that do not fit the pattern as syntactic outliers. For example, if the attribute is "blood type", the recognized pattern could be *one or two uppercase letters followed by a + or a - sign*; then, "ABO" would be considered a syntactic outlier. (*ii*) Repeated Pattern Identification singles out values that contain repeated patterns, such as 0101010101 in a 10-digit phone number, or "blablabla" in a text attribute.
- Second, in numerical attributes, **statistical outliers** can be found by leveraging common outlier detection methods [16]. This allows to identify as DMVs, numerical values that do not fit the extent of the other values, e.g., negative values in a distance attribute.

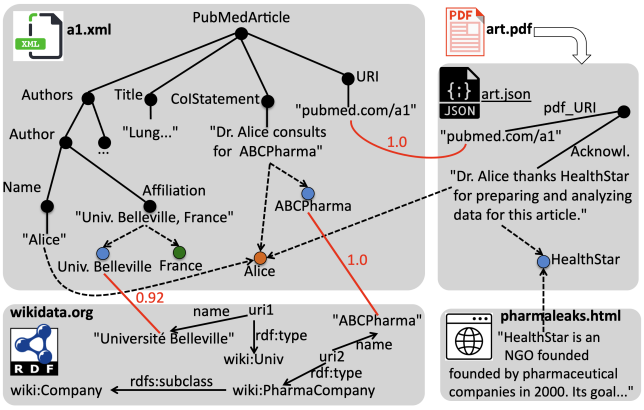

Figure 1: Sample ConnectionLens graph.

- Finally, some **inlier DMVs**, called **Random DMVs** in [21] can be identified. These are legal attribute values, which do not stand out as outliers; they are the hardest to find even for an application domain specialist. The "Alabama" example from Section 1 is a typical example. Inlier DMVs are detected in [21] under the MAR and MCAR hypotheses; the authors state that detecting DMVs under the NMAR model is hard to impossible. The intuition being exploited is that *DMVs are frequent values* (because the lack of information is assumed to occur more frequently than an actual, correct value). Thus, one must find amongst the most frequent values, those which are DMVs. To this purpose, each frequent value is successively replaced by an actual null. If, by doing this, the (original and introduced) null values follow the MAR or MCAR models, then we consider that value as a good DMV candidate. Then, the MEUS method from [17] is applied to each candidate to detect the DMVs.

The FAHES team has developed a tool using these methods to detect all three types of DMV in a relational database.

Finding disguised nulls is one among the many problems raised by poor *data quality*, problems which have been traditionally addressed through *data cleaning*. Data quality raises many real problems, which to this day still needs solutions, Traditional approaches for data cleaning were rule-base [13, 14, 22]. Newer techniques are now based on machine learning approaches, e.g., [18]. Our work is part of this effort to improve data quality. Disguised null detection can be seen as a data cleaning task; it is also related to data profiling [1], since DMV detection also allows to characterize a certain attribute (set of nodes) by the percentage of their values which are disguised nulls.

3 MOTIVATING EXAMPLE

We loaded 400.000 **PubMed bibliographic notices** in a ConnectionLens graph, out of which we extracted (paper ID, conflict of interest statement) pairs. These CoI statements cover any kind of benefits (funding, personal fees etc.) that authors report with various organizations such as companies, foundations etc.

In Figure 1, "Dr. Alice consults for ABCPharma" (in the upper left) is such a conflict of interest, part of the XML bibliographic notice; "Dr. Alice thanks HealthStar... this article" (at the top right) is another one. PubMed data originates from various biomedical journals. Some do not provide CoI information; in this case, the CoI is an empty string. Others provide a default disguised null value, e.g., "The authors report no conflict". Finally, some journals only allow free text, leading to a large variety of disguised nulls.

ConnectionLens extracts named entities from all text nodes, regardless of the data source they come from, using trained language models. In the figure, blue, green, and orange nodes denote Organization, Location, and Person entities, respectively. Each entity node is connected to the text node it has been extracted from, by an extraction edge, which also records the confidence (between 0 and1) of the extractor. Finally, nodes are compared to find that some may be equivalent (solid red edges) or similar(dashed red edges).The original motivation of this work was to avoid connecting, by such equivalence or similarity edges, two identical or similar values,if one of them is a disguised null, since this would lead to paths in the graph that have no meaning.

Furthermore, extracting entities is really time consuming and with disguised null values, we are extracting entities over values we cannot exploit, thus losing time unnecessarily.

4 DETECTING DISGUISED NULLS WITH ENTITY PROFILES

ConnectionLens [5–7, 10] integrates heterogeneous data into a graph. Figure 1 (reused from [5]) illustrates this in a data journalism scenario proposed by investigative journalists with whom we collaborate: we integrate four datasets (delimited by gray areas) in order to build a comprehensive database of information about conflicts of interest in the biomedical domain. In this example, we use a PubMed XML bibliographic notice, the corresponding medical article (in PDF) transformed into a JSON document, an RDF fragment from Wikidata, and an HTML page such as those set up by journalists on TobaccoTactics, DesmogBlog etc., to share information about organizations such as industry lobby groups.

After inspecting DS3 and trying to find what distinguishes an actual conflict of interest (CoI, in short) from a disguised null, we made the following observation. An actual CoI (such as those involving Alice in Figure 1) is either of the form "Researcher A was funded by B", or of the form "The authors acknowledge funding from C". Thus, a person name may be present (in other cases, we just find "The authors"), but an organization is always involved. Thus, we can say that the *entity profile* of a CoI is: it must contain at least an organization.

This leads to the following disguised null detection method:

- Extract all named entities from the CoI strings (through regular ConnectionLens data ingestion);
- Declare those CoI strings in which no Organization entity was found, as disguised nulls.

The accuracy of this method is exactly that of the entity extractor; it has been shown in [3, 4] (for English) and in [5] (for French) that the accuracy is quite high. Its drawback is that *extracting entities from all CoI strings is very lengthy*. This motivates the search for a faster technique, which, on one hand, could identify the DMVs, while at the same time also reducing the entity extraction (thus, the actual ConnectionLens graph creation) time.

In practice, of course, we only extract entities once from each distinct string. It turns out that DS3 had a high number of duplicates (especially some very popular disguised nulls). Removing duplicates from DS3 leads to a new dataset we denote **DDS3**, consisting of 82.388 values. We use this de-duplicated dataset for the DMV detection methods described below.

5 DISGUISED NULL DETECTION THROUGH EMBEDDING AND CLASSIFICATION

With entity extraction, we have a good method to detect disguised nulls. However, this method is very time-consuming, so we need to find a faster method.

Our initial approach has been to *cluster* the values, in order to obtain DMV cluster(s) separated from non-DMV clusters. In particular, we experimented with the K-means [16] algorithm, setting the number of clusters to 10. We could indeed see that DMVs "mostly" clustered together, but the separation was not very good.

Thus, we looked for an alternative method. Our idea is: we could extract entities from a small part of the data, then train a Machine Learning model to recognize DMVs (based on the method from the previous section), and finally use this model to predict whether a yet-unseen value is a DMV or not.

5.1 Textual data representation

To classify our string (text) values, we needed to represent them as multi-dimensional numeric vectors. First, we apply a set of common text pre-processing: suppressing punctuation, normalization and stemming.

Then, we need to project the pre-processed texts in a multidimensional space. Many techniques can be used for this purpose. Transformers like BERT [12] have been proven really efficient for many Natural Language Programming (NLP). However, experimenting with BERT and similar tools has shown that obtaining a sentence embedding [23] is time-consuming, which does not suit our time saving objective.

Instead, we opted for the well-known TF-IDF (Term-Frequency - Inverse-Document-Frequency) representation, commonly used in Natural Language Processing. TF-IDF weights term frequencies in each document according to the frequency of the term across the corpus. Intuitively, TF denotes that if a word occurs many times in a document, its relevance should be boosted as it must be more meaningful as it appears frequently. Conversely, IDF stands for the fact that if the word appears frequently in many documents, then it is just a frequent word and its relevance should be decreased. We finally keep only the top 20.000 terms with highest TF-IDF score to reduce dimensions.

5.2 Classification model

To classify texts as DMVs or non-DMVs, we decided to rely on a Random Forests classifier [9]. These classifiers are not the fastest, but they are quite efficient over complex data. Other classifiers might work as well; our goal here is to investigate whether the above approach, which trains the classifier on extraction results, can provide a more efficient DMV detection method, by avoiding to extract entities from a certain part of the input.

As we show in Section 6.5, this is indeed the case; even for small training set sizes (that is, even if entities are fully extracted only from a small part of the data), the classifier learns to predict quite accurately DMVs, while sparing significant entity extraction time.

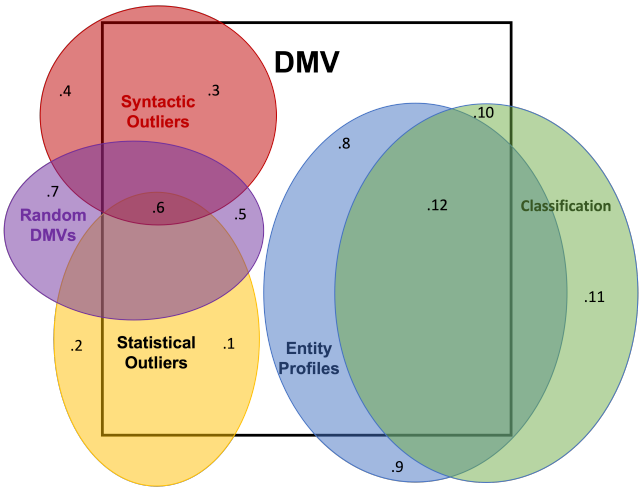

Figure 2: DMVs and the scope of each DMV technique

5.3 Discussion

Our methods detect types of DMVs that are different from the ones detected by FAHES. Figure 2 represents a diagram showing how each method detects DMVs. FAHES techniques work better over structured data, whereas our techniques work over free-texts. Here we will detail with examples what DMVs or wrongly detected DMVs are in each part of the diagram:

(1) Correctly detected statistical outliers, e.g.: In a human heights attribute, a height of 3 meters.
(2) Wrongly detected statistical outliers, e.g.: In a dataset containing salaries of the employees of a company, if the CEO salary is 10 times higher than any of his employees, this could be wrongly detect as a DMV.
(3) Correctly detected syntactic ouliers, e.g.: In a blood type attribute, the value *'ABO'*.
(4) Wrongly detected syntactic outlier, e.g.: In a name attribute, *François-Noël* which is a composed name is detected as syntactic outlier because of the *'-'*, whereas it is a valid name.
(5) Correctly detected Random DMVs, e.g.: A default value such as *Alabama* for a state, detected thanks to the MEUS technique.
(6) A DMV can be at the same time a syntactic outlier, a random DMV and a statistical outlier, e.g., a default distance value of -1.
(7) Wrongly detected random DMVs. In a poll where we ask for favourite colors, *blue* might come really often. Lacking correlation with other attributes, *blue* could be wrongly detected as a random DMV.
(8) Correctly detected DMV with entity profile technique, e.g.: In a conflict of interest paragraph : *"The authors report no conflict of interest."*
(9) Wrongly detected DMV with entity profile technique. These are errors of the entity extractor when it misses an organization.

(10) Correctly detected DMV with the classification method that was not detected by the entity extraction method. For example, in *'John McDonalds declares no conflict of interest'*, the entity extractor could detect McDonalds as an Organisation, and classify the value as an actual CoI instead of a DMV.
(11) Wrongly detected DMV with the classification method, these are errors of the classifier.
(12) Most of the DMV detected by the entity profile technique are as well detected by the classification technique.

6 EXPERIMENTAL EVALUATION

We now describe performance-oriented experiments with the disguised null detection techniques described above.

6.1 Datasets

To conduct our experiments, we have built three ConnectionLens graphs out of real-world datasets:

(1) We have loaded the most complete **HATVP XML** transparency dataset (35MB), with data about 270.000 people, in a ConnectionLens graph. From this, we extracted the *montant* (monetary amount) fields which appeared to contain many disguised nulls[1].
(2) We loaded a smaller **HATVP CSV** dataset (2.1 MB), containing information about 9.000 people; this dataset is relational-looking, which simplifies processing it through FAHES.
(3) We loaded 400.000 **PubMed bibliographic notices** in a graph, out of which we extracted (paper ID, conflict of interest statement) pairs. These CoI statements cover any kind of benefits (funding, personal fees etc.) that authors report with various organizations such as companies, foundations etc. In Figure 1, "Dr. Alice consults for ABCPharma" (in the upper left) is such a conflict of interest, part of the XML bibliographic notice; "Dr. Alice thanks HealthStar... this article" (at the top right) is another one. PubMed data originates from various medical journals. Some do not provide CoI information; in this case, the CoI is an empty string. Others provide a default disguised null value, e.g., "The authors report no conflict". Finally, some journals only allow free text, leading to a large variety of disguised nulls. We will denote these datasets **DS1**, **DS2** and **DS3**, respectively.

6.2 Settings

All experiments were performed on a MacBook Pro 16 inches from 2019, with a 2.4 GHz Intel Core i9 8-core processor and 32 GB 2667 MHz DDR4 memory. We used ConnectionLens[2] to build the graphs, including in particular the extraction of named entities using Flair [3, 4], which we had retrained for French [7]. ConnectionLens graphs are stored in Postgres 9.6; experiment code was written in Python 3.6.

6.3 Null detection through FAHES

We have appplied FAHES [21] on the three datasets described previously. Next we will comment on the results obtained by the null detection through FAHES.

[1]The transparency entry forms require filling in the worth of participations or ownerships in various companies; companies which have closed or did not make benefits, or pro-bono activity, lead to disguised nulls.
[2]Available from https://gitlab.inria.fr/cedar/connectionlens

Table 1: Entity extraction method times

Values	Total characters	Extraction times (s)
500	163.203	56
5.000	1.604.141	669
10.000	3.281.345	1.320
15.000	5.000.364	2.175
20.000	6.728.493	2.620

DS1 Among the 270.000 values of the numeric *amount* attribute, FAHES correctly found the disguised null 0 (45.000 occurrences) as Random DMVs (Inliers). It also detected 372.2196 (4 occurrences) as a numerical outlier DMV; this is wrong. All the amount values are numbers, and as far as we could see, there are no other disguised nulls.

DS2 In this relational dataset, in an attribute entitled `filename`, FAHES identified **correctly** the disguised null *dispense* (120 occurrences) as a Random DMV. Then, FAHES identified **wrongly** other values as being disguised nulls, in all cases as Syntactic Outliers DMV. The values falsely flagged as disguised nulls are:

- *François-Noël* (3 occurrences) in the attribute `given name`;
- *BÉRIT-DÉBAT* (6 occurrences) and *KÉCLARD-MONDÉSIR* (3) in the attribute `name`;
- *di* (4480 occurrences) in the attribute `document type`; this is the acronym for *déclaration d'intérêt*;
- *2A* and *2B* as departement numbers; they are, in fact, correct numbers of French departments in Corsica;
- four distinct, correct URLs within the `photo_url` attribute, probably because their structure did not ressemble the others'.

DS2 seems to include no other disguised nulls.

Results on DS3 Out of the 400.000 values, FAHES correctly identified *The authors have declared that no competing interests exist* (31.891 occurrences) as a Random DMV. However, visual inspection exhibited many other disguised nulls (we will revisit this below). FAHES fails to find them because freely written texts rarely coincide, thus FAHES' statistical approach based on value frequencies considers many disguised nulls rare (thus non-null), which is wrong.

Experiment conclusion FAHES [21] performs quite well with numbers but is less convincing when it handles textual data. Indeed, DMVs detected as Syntactic outliers are often false positives. With a bit of domain knowledge it is possible to manually discard these DMVs, however, it shows the limits of the methods used to detect DMVs as syntactic outliers. FAHES has also shown its limitations by missing many DMVs on long free text data, which shows the need of new methods to treat these cases.

6.4 Disguised null detection through entity profiles

To measure performances of the entity profiles technique, we performed 5 experiments with respectively the 500, 5.000, 10.000, 15.000 and 20.000 first values of dataset DDS3 (introduced at the end of Section 4). The objective here is to measure the extraction time as a function of the input size; this gives an indication of the time needed to extract disguised nulls using the Entity Profile

Table 2: Impact of training set size over the performances of disguised null detection

Values in Training-set	16.477	8.238	823
Extraction Times (s)	2.153	1.075	108
Training Times (s)	54	23	6
Prediction Times (s)	10	10	11
Total Times (s)	2.217	1.108	125
Precision	0,939	0,933	0,885
Recall	0,948	0,946	0,942
F1-score	0,943	0,940	0,913

method. We present the results in Table 1. We observe that extracting entities is time consuming and that the extraction time is almost linear to the number of values. For the complete dataset DDS3, we can expect to have an extraction time around 11.000 seconds (183 minutes), which is quite lengthy.

6.5 Disguised null detection through embedding and classification

The most common training-test split method consists on separating the dataset with 20% used for training and 80% for testing. We know that in our case, the most time-consuming operation is to label the training set with the entity extraction technique. To gain time, we want thus to reduce the training set.

To evaluate the impact of the training set size on the performance of the model used to detect disguised nulls over DDS3, we have performed 3 experiments. We trained models with respectively 20% (16477 values), 10% (8238 values) and 1% (823 values) of the dataset and report the comparison of the performances of each model in Table 2. In this table, the precision, recall (and thus also F1) are computed using the result of the entity profile method (Section 4) as gold standard; as we have seen, it is a time-consuming technique. Table 2 shows that we can attain very good precision, even if the model is trained on a small part of the dataset, while saving significant amounts of time. In the context of our project, recall is more important than precision, since decreasing recall means losing valuable information (potential CoIs) while decreasing precision means extracting entities from useless values (strings which do not contain any), thus losing some time. Our experiments show that in our problem, recall is less sensitive than precision to the reduction of the training-set size which suits our purpose.

With respect to our motivating example, building the graph for DDS3 took around 11.000 seconds. Using our method with a training-set of 1% of the values (823 values) takes now the time to predict on which values we have to apply the extractor (125 seconds), to which we add the time to extract the valuable values. We have found on our dataset that there are around 45.000 valuable values. We need 5.900 seconds to extract those. That brings us to a total of **6.000 seconds to build our graph instead of 11.000 seconds previously** without losing much information.

7 CONCLUSION

In this work, we exposed the limits of the existing DMV techniques with textual and heterogeneous data. A first technique we studied is to extract entities from each value of the database and exploit so-called *entity profiles* (expected entities in a non-null value) to identify disguised nulls. While highly accurate, this is expensive time-wise, because of the extraction. For efficiency, instead, we trained a classification model (Random Forest), only with partial samples of our dataset, labeled it with entity profiles, and predict (classify) the other values as DMVs or not. This technique saves significant extraction time, while also having very good accuracy.

As part of our future work, we could also try to optimize the hyper-parameters of the classification model, in order to further increase its performance.

REFERENCES

[1] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. *Data Profiling*. Morgan and Claypool, 2020.

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art NLP. In *ACL*, 2019.

[4] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *ACL*, 2018.

[5] Angelos Christos Anadiotis, Oana Balalau, Catarina Conceicao, Helena Galhardas, Mhd Yamen Haddad, Ioana Manolescu, Tayeb Merabti, and Jingmao You. Graph integration of structured, semistructured and unstructured data for data journalism. *Information Systems*, 2021.

[6] Angelos-Christos G. Anadiotis, Oana Balalau, Théo Bouganim, Francesco Chimienti, Helena Galhardas, Mhd Yamen Haddad, Stephane Horel, Ioana Manolescu, and Youssr Youssef. Empowering investigative journalism with graph-based heterogeneous data management. *IEEE Data Engineering Bulletin*, September 2021.

[7] Oana Balalau, Catarina Conceição, Helena Galhardas, Ioana Manolescu, Tayeb Merabti, Jingmao You, and Youssr Youssef. Graph integration of structured, semistructured and unstructured data for data journalism. BDA, October 2020.

[8] Raphaël Bonaque, Tien Duc Cao, Bogdan Cautis, François Goasdoué, Javier Letelier, Ioana Manolescu, Oscar Mendoza, Swen Ribeiro, Xavier Tannier, and Michaël Thomazo. Mixed-instance querying: a lightweight integration architecture for data journalism. In *VLDB*, 2016.

[9] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] Camille Chanial, Rédouane Dziri, Helena Galhardas, Julien Leblay, Minh-Huong Le Nguyen, and Ioana Manolescu. ConnectionLens: Finding connections across heterogeneous data sources (demonstration). *PVLDB (also at BDA)*, 11(12), 2018.

[11] Edgar F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13, June 1970.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2019.

[13] Helena Galhardas, Daniela Florescu, Dennis E. Shasha, and Eric Simon. Declaratively cleaning your data with AJAX. In Anne Doucet, editor, *BDA*, 2000.

[14] Helena Galhardas, Daniela Florescu, Dennis E. Shasha, Eric Simon, and Cristian-Augustin Saita. Declarative data cleaning: Language, model, and algorithms. In *VLDB*, 2001.

[15] Joachim Hammer, Hector Garcia-Molina, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. Information translation, mediation, and mosaic-based browsing in the TSIMMIS system. In *SIGMOD*, 1995.

[16] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining Concepts and Techniques*. Morgan Kaufmann, 2011.

[17] Ming Hua and Jian Pei. Cleaning disguised missing data: A heuristic approach. In *SIGKDD*, 2007.

[18] Ihab F. Ilyas and Mohamed A. Soliman. *Probabilistic Ranking Techniques in Relational Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[19] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 1987. First edition.

[20] Ronald K. Pearson. The problem of disguised missing data. *SIGKDD Explor. Newsl.*, 8(1):83–92, June 2006.

[21] Abdulhakim A. Qahtan, Ahmed Elmagarmid, Raul Castro Fernandez, Mourad Ouzzani, and Nan Tang. Fahes: A robust disguised missing values detector. In *SIGKDD*, 2018.

[22] Vijayshankar Raman and Joseph M. Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, 2001.

[23] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.